

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-083310

(43)Date of publication of application : 31.03.1998

(51)Int.Cl.

G06F 9/445
G06F 9/06
G06F 12/14

(21)Application number : 09-151768

(71)Applicant : INTERNATL BUSINESS MACH
CORP <IBM>

(22)Date of filing : 10.06.1997

(72)Inventor : DAN ASIT
RAMASWAMI RAJIV
SITARAM DINKAR

(30)Priority

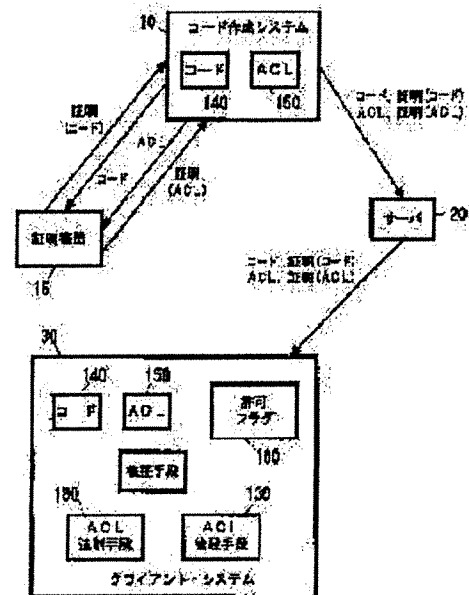
Priority number : 96 661517 Priority date : 11.06.1996 Priority country : US

(54) PROGRAM CODE DISTRIBUTING METHOD AND ITS SYSTEM

(57)Abstract:

PROBLEM TO BE SOLVED: To provide an authentication system for allowing a relied third party to confirm the author of a program and to sign a certification for guaranteeing the perfection of the program.

SOLUTION: A program code 140 is capsuled together with a guarantee and access control list(ACL) 150. ACL 150 describes an allowable condition and a resource required by the code 140. A forcing mechanism assigns the allowable condition of a system and a resource according to ACL 150. For example, a code preparing system 10 communicates with a certifying organization 15 being the relied third party. The organization 15 issues the certificate of the code 140 and the certificate of ACL 150 of the code 140. Once the certificate is issued, nobody can change the code 140 and ACL 150 without invalidating the certificate. The code 140, its ACL 150 and their certificates are stored in a server.



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平10-83310

(43) 公開日 平成10年(1998) 3月31日

(51) Int.Cl.*	識別記号	序内整理番号	F I	技術表示箇所
G 0 6 F 9/445			G 0 6 F 9/06	4 2 0 J
9/06	5 5 0			5 5 0 Z
12/14	3 1 0		12/14	3 1 0 K
				3 1 0 Z

審査請求 未請求 請求項の数18 O L (全 16 頁)

(21) 出願番号	特願9-151768	(71) 出願人	390009531 インターナショナル・ビジネス・マシーンズ・コーポレーション INTERNATIONAL BUSINESS MACHINES CORPORATION アメリカ合衆国10504、ニューヨーク州 アーモンク (番地なし)
(22) 出願日	平成9年(1997) 6月10日	(72) 発明者	アスイット・ダン アメリカ合衆国10604、 ニューヨーク州 ウェスト ハリソン ゲインズボーク アヴェニュー 75
(31) 優先権主張番号	08/661517	(74) 代理人	弁理士 坂口 博 (外1名)
(32) 優先日	1996年6月11日		
(33) 優先権主張国	米国 (US)		

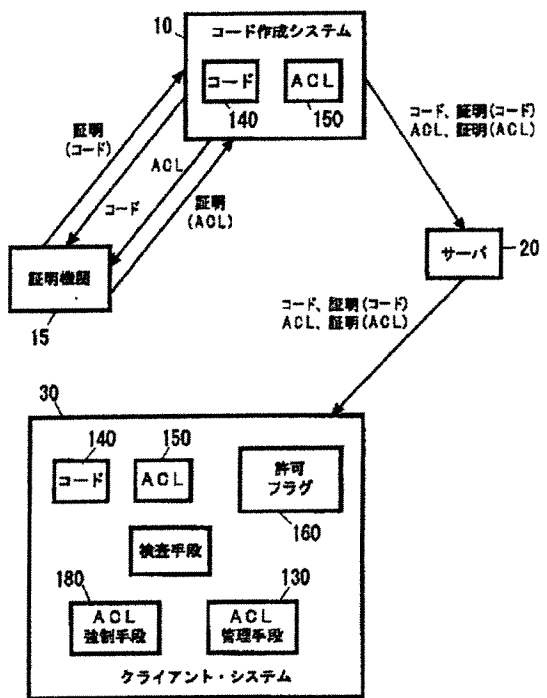
最終頁に続く

(54) 【発明の名称】 プログラム・コードの配布方法及びシステム

(57) 【要約】

【課題】 信頼される第三者がプログラムの作者を確認しプログラムの完全性を保証するための証明に署名する認証方式を提供すること。

【解決手段】 プログラム・コードは証明およびアクセス制御リスト (ACL) と共にカプセル化されている。アクセス制御リストはコードによって要求される許容条件とリソースを記述する。強制メカニズムはアクセス制御リストに従ってシステムの許容条件とリソースを割り振る。実施例において、コード作成システムは、信頼される第三者である証明機関と通信する。証明機関はコードの証明と、そのコードのアクセス制御リストの証明とを発行する。一度、証明が発行されると、いかなる者もその証明を無効にすることなくコードまたはアクセス制御リストを変更することはできない。コード、そのアクセス制御リスト、およびそれらの証明はサーバ中に記憶される。



【特許請求の範囲】

【請求項1】プログラム・コードの配布方法であって、信頼される第三者の証明を受け取りシステムへ提供するステップを含み、上記プログラム・コードの検査された無害の動作に必要なリソースと許容条件を、上記信頼される第三者の証明の中にコンピュータで読み取り可能に記述したことを特徴とする、プログラム・コードの配布方法。

【請求項2】請求項1において、上記証明をプログラム・コードと共にカプセル化するステップを含む、プログラム・コードの配布方法。

【請求項3】請求項1において、上記受け取りシステムが上記証明を読み取るステップと、上記証明中に指定された許容条件を超過しないように上記受け取りシステムのリソースと許容条件を割り振るステップとを含む、プログラム・コードの配布方法。

【請求項4】請求項2において、上記証明に関連してユーザが選択したオプションに従って、上記受け取りシステムのリソースに対する上記プログラム・コードのアクセスと許容条件を否定または肯定するステップを含む、プログラム・コードの配布方法。

【請求項5】請求項1において、コンピュータで読み取り可能に記述した上記リソースと許容条件が暗号形式で上記受け取りシステムへ提供されることを特徴とする、プログラム・コードの配布方法。

【請求項6】請求項1において、暗号化された検査データが上記証明の中に含まれており、上記受け取りシステムが上記検査データを解読して上記リソースと許容条件の記述の完全性を検査することを特徴とする、プログラム・コードの配布方法。

【請求項7】請求項3において、必要なリソースの記述の中に、上記プログラム・コードによって使用される各リソースの量と、そのリソースの最大許容消費率の両方が記述されていることを特徴とする、プログラム・コードの配布方法。

【請求項8】請求項3において、必要な許容条件の記述の中に、上記プログラム・コードによってアクセスされる上記受け取りシステムの特定の機能が記述されていることを特徴とする、プログラム・コードの配布方法。

【請求項9】請求項1において、上記プログラム・コードの機能性が、上記第三者による検査に従って上記第三者の証明の中に記述されていることを特徴とする、プログラム・コードの配布方法。

【請求項10】請求項1において、上記プログラム・コードが、サーバからプログラム・オブジェクトとしてダウンロードされるアプレットであることを特徴とする、プログラム・コードの配布方法。

【請求項11】請求項1において、上記受け取りシステムの中で上記プログラム・コードの異なったユーザに対しては異なったリソースと許容条件を割り振ることを特

徴とする、プログラム・コードの配布方法。

【請求項12】請求項11において、上記異なったユーザに許されるリソースと許容条件の組が、上記プログラム・コードのインストール時に決定されることを特徴とする、プログラム・コードの配布方法。

【請求項13】請求項11において、上記異なったユーザに許されるリソースと許容条件の組が、上記プログラム・コードの実行時に決定されることを特徴とする、プログラム・コードの配布方法。

【請求項14】プログラム・コードの配布方法であって、信頼される第三者によるプログラム・コードの証明の中に、該プログラム・コードの検査された無害の動作に必要なリソースと許容条件をコンピュータで読み取り可能に記述したものを、その証明を上記プログラム・コードと共にカプセル化して暗号形式で受け取りシステムへ提供するステップと、上記受け取りシステムが上記暗号形式の証明を読み取るステップと、上記読み取られた証明の完全性を上記受け取りシステムによって決定するステップと、上記完全性が検査された後に、上記証明の中で指定された許容条件を超過しないようにユーザが選択したオプションに従って上記受け取りシステムのリソースと許容条件を割り振るステップと、上記割り振りに従って上記プログラム・コードを実行するステップとを含み、上記必要なリソースの記述の中には、上記プログラム・コードによって使用される各リソースの量とその最大許容消費率との両方が記述され、上記許容条件の記述の中には、上記プログラム・コードによってアクセスされる上記受け取りシステムの特定の機能が記述されていることを特徴とする、プログラム・コードの配布方法。

【請求項15】コンピュータ・システムへプログラムとデータを移入する移入装置と、上記コンピュータ・システムの動作を制御するオペレーティング・システムと、コードの検査された無害の動作に必要なリソースをコンピュータで読み取り可能に記述したものを、上記データから抽出してそれを所与のプログラムに関連づけるアクセス・ロジックと、該アクセス・ロジックに含まれて、上記コンピュータで読み取り可能に記述したものの完全性を示す検査データを生成する完全性検査ロジックと、上記オペレーティング・システムに接続され、上記検査データにตอบสนองして多数のリソースの各々について上記受け取りシステムの中で消費量と消費率を追跡し割り振って、上記記述の中に指定された割り振りを超過しないようにする強制ロジックと、上記割り振りに従ってプログラム・コードを実行するプロセッサと、を具備するコンピュータ・システム。

【請求項16】請求項15において、ランダム・アクセス・メモリに記憶されたデータ構造が上記強制ロジックに接続されており、該データ構造の中には、消費された実際のリソースを追跡する第1のフィールドと、リソー

スの消費率を追跡する第2のフィールドと、上記記述から引き出されたりソース消費の限度を記憶する第3のフィールドと、上記記述から引き出されたりソース消費率の限度を記憶する第4のフィールドとが含まれていることを特徴とする、コンピュータ・システム。

【請求項17】請求項15において、上記アクセス・ロジックの中に、プログラム・コードを含むパッケージから上記記述を非カプセル化する手段と該記述を解釈する手段とが含まれていることを特徴とする、コンピュータ・システム。

【請求項18】請求項15において、上記強制ロジックの中に、上記記述に関連してユーザが選択したオプションに従ってコンピュータ・システムのリソースに対するコード・アクセスと許容条件を否定または肯定する手段が含まれていることを特徴とする、コンピュータ・システム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、一般的には、ネットワークのような頒布システムを使用してソフトウェアを送付することに関する。

【0002】

【従来の技術】ソフトウェアの頒布業は最近の大きな産業である。今日、ソフトウェアはディスクットおよびCD-ROMを介して頒布され、また次第にネットワークを介して頒布されるようになった。さらに、最近では、単にインターネットを介して遠隔サイトからコードをダウンロードするだけでなく、クライアントはサーバからアプレットをダウンロードして実行することができる。これは、ジャバ(Java)・プログラミング言語、またはテレスクリプトのような他の言語で提案されているパラダイムである。

【0003】他から得られたコードに伴う重要な問題は、セキュリティの問題である。たとえば、UNIXオペレーティング・システムでは、コードはクライアントのすべての特権を具備しているクライアント・シェルの中で実行される。そのような特権には、クライアントのすべてのファイルにアクセスすること、メールを送ること、さらには不法侵入を試みることも含まれる。ジャバは、このようなアプレットの問題を解決するために、非常に制限された環境でアプレットを実行する。その結果、ジャバのアプレットの有用性と機能性は制限される。他方、ジャバのアプリケーションはオペレーティング・システムによって提供されるセキュリティに依存し、どのような認証(authentication)にもなじまない。したがって、それらアプリケーションは他のコードと同じようなセキュリティの問題を提起する。

【0004】認証の1つの形式は、「Trusted Distribution of Software O

ver the Internet」(A. D. Rubin, Internet Society Symposium on Network and Distributed Security, 1995)に提案されている。そこでは、信頼される第三者が証明に署名して、プログラムの作者を確認するとともにプログラムの完全性を保証している。

【0005】

【発明が解決しようとする課題】この方法によって、クライアントはコードの認証を検証できるが、この方法は、コードに関連した許容条件を柔軟に指定するものではなく、またクライアントがこれらの許容条件を自動的に強制する方法を提供するものでもない。

【0006】

【課題を解決するための手段】本発明は、信頼される第三者が証明に署名して、プログラムの作者を確認するとともにプログラムの完全性を保証するものであるが、プログラム・コードはカプセル化されている(すなわち、証明およびアクセス制御リストと関連づけられている)。アクセス制御リストはコードによって要求される許容条件とリソースを記述している。さらに、本発明は、アクセス制御リストに従ってシステムに許容条件とリソースを割り振る強制メカニズムを提供する。

【0007】実施例において、コード作成システムは、信頼される第三者である証明機関と通信する。証明機関はコードの証明と、そのコードに関するアクセス制御リストの証明を発行する。一度、証明が発行されると、いかなる者も、その証明を無効にすることなしに、そのコードまたはアクセス制御リストを変更することはできない。コードおよびそのアクセス制御リストは、それらの証明と共にサーバに記憶される。コードまたはアクセス制御リストをダウンロードするクライアントは、コードないしアクセス制御リストの完全性を検査することができ、システムは許容条件とリソースを超過しないようにアクセス制御リストを実施することができる。

【0008】

【発明の実施の形態】図1は、本発明に従うコード送付および証明システムのブロック図である。そこには1つまたは複数のコード作成システム10、証明機関15、1つまたは複数のサーバ20、および1つまたは複数のクライアント・システム30が示される。クライアント・システムは通常の広域ネットワーク(WAN)またはローカル・エリア・ネットワーク(LAN)を介してサーバへ接続される。コード作成システムは、証明機関によって証明してもらいたいプログラム・コード140と、そのコードに関するアクセス制御リスト(ACL)150を含む。証明機関は、クライアント・システムによって広く使用され知られている公用キーKを提供する。さらに、証明機関は、それだけが知っている私用キーPを保持している。コードの証明を提供するために

は、証明機関はコード名とコードの暗号ハッシュ (hash) を含む証明を作成し、私用キーを使用してそれに署名する。今や、その証明は証明機関の署名を無効にすることなしに変更することはできない。同様に、証明機関は、コードに関連したアクセス制御リスト用の証明にも署名する (もし望むなら、コードとそのアクセス制御リストの双方のために、ただ1つの証明にサインするようにしてもよい)。図1の実施例では、クライアント・システムは、WANまたはLANを介して証明、アクセス制御リスト、およびコードを受け取る。しかし、その方法に替えて、クライアント・システムは、取り外し可能記憶媒体を介して、証明されたコードを受け取ることができる点に注意されたい。このような記憶媒体は、フロッピー・ドライブまたは光学ディスク・ドライブのようなローカル移入装置によって読み取られる。

【0009】図2に示されたクライアント・システムは検査手段110、アクセス制御リスト (ACL) 管理手段130、実行手段170、アクセス制御リスト (ACL) 強制手段180、およびクライアント・インタフェース190を含む。さらに、クライアント・システムは通常のCPUおよび関連制御ロジック162、通信コントローラ/ネットワーク・インタフェース194、CD-ROM/フロッピー・ディスク・サブシステム196、および他のリソース (たとえば、ネットワーク・アクセス・モジュール、ディスプレイ・サブシステム、および各種の特殊目的アダプタ) を含む。当業者にとって、クライアント・システムが他の多くの通常のコンポーネントを含むことは明らかであろうが、それらについては、ここで詳細に説明しない。

【0010】アクセス制御リスト管理手段130とアクセス制御リスト強制手段180は、プログラム・コードとして実施するのが望ましい。その動作については後で詳細に説明する。実行手段はクライアントのオペレーティング・システム (図示されず) の通常の部分であり、クライアント・システム上で、移入されたコードを実行するために使用される。クライアント・インタフェース190は画面グラフィカル・ユーザ・インタフェースのフロントとして実施することができ、アクセス制御リスト管理手段130とクライアントとの連絡を可能にする。(たとえば、それによって、クライアントは、指定されたリソースへのプログラム・アクセスを許すか許さないかをアクセス制御リスト管理手段130に命令したり、プログラム・アクセスを管理したりすることができる。) 検査手段110はプログラム・コードとして実施することが望ましい。それは解読モジュールを含み、アクセス制御リストと共に移入されたコードの認証を検査する。さらに、検査手段はハッシング (hashing) モジュールを含む。このモジュールは、移入されたコードとアクセス制御リストの完全性をチェックする。

【0011】「コード/アクセス制御リストおよび証

明」100がダウンロードされると、まず検査手段110は証明の上にある証明機関のサインが正しいかどうかを検査する (証明機関の既知の公用キーを使用する)。次に、検査手段はコード/アクセス制御リストの暗号ハッシュを計算して、それが証明中の値と一致するかどうかを検査する。サインが正しくないとき (すなわち、ハッシュが一致しないとき)、そのコードとアクセス制御リストは拒絶される (拒絶ステップ120)。検査がOKであれば、アクセス制御リスト管理手段130が呼び出される。アクセス制御リスト管理手段はクライアント・インタフェース190を介してクライアントへアクセス制御リスト (後で説明する) を表示し、クライアントがアクセス制御リスト内の個々の項目を許可するか許可しないかを確かめる。アクセス制御リスト管理手段130は、クライアント・インタフェースを介してクライアントから命ぜられたとおりに、コード140を記憶し、許可フラグ160と共にアクセス制御リスト150を記憶する。

【0012】アクセス制御リスト強制手段180によってアクセスを制御することのできるリソースには、ファイル・システム、特定のファイル、およびシステム呼び出しのような論理リソースと、ディスク・スペース、ディスク・アクセス、メイン・メモリの割り振り、および各種のシステム・コントローラやアダプタへのアクセスのような物理リソースがある。論理リソース (クライアント・システムの機能) へのアクセスについては、個々の項目が許可されるか許可されないかを、許可フラグ160が表示する。物理リソースへのアクセスについては、最大許容量または最大許容消費率を示すために許可フラグを使用できる。

【0013】さらに、図2はクライアント・システムの異なった部分がどのように相互に動作するかを示す。複数ユーザ・クライアント・システムでは、各ユーザがそれぞれの許可フラグ群を持つことができる。さらに、アクセス制御リストの中には環境変数を入れることができる。実行中に環境変数を変えることによって、個々のユーザはアクセス特権をカスタマイズすることができる。アクセス制御リストと許可フラグはセキュリティ領域に記憶され、この領域の読み出しと更新はアクセス制御リスト強制手段180に許される特権である。

【0014】アクセス制御リスト管理手段は、アクセス制御リストの許可条件を表示または変更するためにクライアントによって任意の時点で呼び出されてよい。コードは実行手段170によって実行される。リソースへのアクセスを許す前に、実行手段はアクセス制御リスト強制手段を呼び出して、アクセスの有効性をチェックする。これは、実行手段170が、アクセス制御リスト強制手段を呼び出す検査ルーチンへのトラップをコードの中に挿入するという方法によって達成される。アクセス制御リスト強制手段の動作は、後で詳細に説明する。

【0015】システムでは、必要に応じてコードとそのアクセス制御リストを別個にまたは組み合わせてダウンロードすることができる。たとえば、コード作成システムの方針として、すべてのクライアントへアクセス制御リストを無料で提供するが、コード自体は有料にすることができる。

【0016】図3はアクセス制御リストを示す。アクセス制御リストは2つの部分から構成される。すなわち、コードによって要求される物理リソースを含む物理リソース・テーブル200と、コードによって要求される許可条件と論理リソースを含む論理リソース・テーブル250である。

【0017】物理リソース・テーブル200の中には、各リソースについての行があり、この行の中には物理リソース名205、リソース属性210、最大許容消費率215、および最大許容量220がある。リソース属性210は、物理装置またはリソース・ディスクに複数の属性があるときに使用される。たとえば、記憶装置についてスペースと1/0数があるような場合である。最大許容消費率215と最大許容量220は、リソースと属性の最大許容消費率と最大許容消費量である。

【0018】論理リソース・テーブル250は、コードによって要求される外部ルーチン（論理リソースと呼ばれる）への各呼び出しについて1つの行を含む。各行には、論理リソース名255とパラメータ・リスト260がある。パラメータ・リスト260はパラメータ・エントリ265のリストを指し示す。各パラメータ・エントリ265は、有効なパラメータ範囲の集合を指定する。すなわち、その集合は組み合わせとして有効なパラメータ値の集合である。パラメータ・エントリ265の中には、次のパラメータ・エントリを指し示すnextPEフィールド280も含まれている。各パラメータのパラメータ範囲は2つのフィールドを含む。すなわち、パラメータ・タイプ270と、そのパラメータの有効範囲を指定するパラメータ値275である。ストリング・パラメータについては、パラメータ・タイプ270はSTRであり、パラメータ値275はそのストリングについて有効な形式を指定する正規表現のリストである。整数パラメータについては、パラメータ・タイプ270はINTであり、パラメータ値275は整数範囲のリストである。パラメータ値275の中には環境変数の名前を入れることができ、その場合、実行時にその環境変数の値を置換することが想定されている。

【0019】クライアントの中のアクセス制御リスト強制手段は、アクセス制御リストの中でコードについて指定された許可条件とリソースが提供され、許可条件／リソースの追加が許されないことを確実にする。

【0020】アクセス制御リストの強制は静的であっても動的であってもよい。静的強制では、コードが実行される前に全面的に強制が行われ、コードが実行されてい

る間の強制は必要とされない。動的強制では、コードが実行されている間に強制を行わねばならない。証明機関自体がアクセス制御リストを検査し、コードによる超過が生じないことが保証されれば、アクセス制御リストの強制機能はクライアント・システムで不必要となろう。

【0021】図4は、アクセス制御リスト強制手段180によって使用されるデータ構造を示す。実行時物理リソース・テーブル300には、各リソースについて1つの行がある。リソース名300、リソース属性310、最大許容消費率315、および最大許容量320は、物理リソース・テーブル200の対応するフィールドのコピーである。実際消費率325および実際使用量330の各フィールドは、それぞれ実行時におけるコードの実際の消費率と実際の消費量を追跡するために使用される。実行時論理リソース・テーブル350には、必要とされる各論理リソースについての行がある。実行時論理リソース・テーブルは論理リソース・テーブルのコピーであり、それに許可フラグが追加されている。許可フラグは、パラメータの有効なそれぞれの組み合わせについて、その組み合わせがアクセス制御リスト管理手段によって許可されているか許可されていないかを示す。論理リソース名355は、論理リソース・テーブル250の対応するフィールドのコピーであり、パラメータ・リスト360は実行時パラメータ・エントリ365のリストを指し示す。実行時パラメータ・エントリのパラメータ・タイプ370およびパラメータ値375は、論理リソース・テーブル内の対応するフィールドのコピーである。nextPEフィールド380は次の実行時パラメータ・エントリ365を指し示し、許可フラグ385は、この実行時パラメータ・エントリ365が許可されているか許可されていないかを示すイエスまたはノーに設定されている。さらに、アクセス制御リスト強制手段はコード開始時点395を追跡する。

【0022】図5は、論理リソースの許可条件がどのようにしてアクセス制御リスト強制手段によって強制されるかを示す。このパスは、コードが外部機能への呼び出しを行うとき常に呼び出される。ステップ410で、アクセス制御リスト強制手段180はパラメータの数、パラメータの値、および呼び出されつつある機能の名前の位置を決定する。その実際の方法は処理系によって異なる。たとえば、ジャバでは、これらはオペランド・スタック上に置かれている。ステップ415では、アクセス制御リスト強制手段は、実行時論理アクセス・テーブル350の中で、呼び出されつつある機能の行の位置を決定し、許可フラグがイエスに設定されている最初の実行時パラメータ・エントリ365の位置を決定する。機能名が見つからないか、そのような実行時パラメータ・エントリ365が存在しない場合、アクセス制御リスト強制手段はステップ470へ進み、呼び出しが許されないことを示して終了する。ステップ420では、アクセス

制御リスト強制手段は最初のパラメータの値の位置を決定して、それを現在のパラメータにする。ステップ425では、アクセス制御リスト強制手段は、現在のパラメータの値が実行時パラメータ・エントリ365によって許されているかどうかをチェックする。値が許されていれば、ステップ430で、アクセス制御リスト強制手段はもっとパラメータがあるかどうかをチェックする。さらにパラメータがあれば、ステップ435で、アクセス制御リスト強制手段は現在のパラメータを次のパラメータに設定しステップ425へ戻る。ステップ430で、それ以上のパラメータがないと、アクセス制御リスト強制手段はステップ450へ進み、呼び出しが許される旨を表示して終了する。

【0023】ステップ425で、許容性のテストが失敗すると、アクセス制御リスト強制手段はステップ445へ進む。そこでは、アクセス制御リスト強制手段が実行時論理リソース・テーブル350をチェックして、許可フラグ385がイエスへ設定されている他の実行時パラメータ・エントリを探す。そのような実行時パラメータ・エントリがなければ、アクセス制御リスト強制手段はステップ470へ進み、呼び出しが許されない旨を表示して終了する。そのような実行時パラメータ・エントリ365があれば、アクセス制御リスト強制手段はステップ420へ進む。

【0024】図6は、物理要件がどのようにしてアクセス制御リスト強制手段によって強制されるかを示す。アクセス制御リスト強制手段は、リソースを割り振る前にステップ500で呼び出される。2つのパラメータ、すなわち要求されたリソース量（REQAMT）と消費予想時間（COMPT）が提供される。ディスク1/0については、REQAMTはディスク1/0の数であり、COMPTはディスク1/0が完了する予想時間である。ステップ505で、アクセス制御リスト強制手段は、このリソースについて実行時物理リソース・テーブル300内の行を決定し、最大許容量320が指定されているかどうか、および実際使用量330にREQAMTを加えたものが最大許容量320を超過するかどうかをチェックする。もし超過すれば、それはステップ527で消費が許されない旨を表示する。最大許容量320を超過しないとき、ステップ510で、アクセス制御リスト強制手段はこのリソースの予測消費率を、（実際使用量330 + REQAMT）／（現時点 - コード開始時点395 + COMPT）に従って計算する。ステップ515で、アクセス制御リスト強制手段は最大許容消費率315が指定されているかどうか、および計画された消費率が最大許容消費率315よりも大きいかどうかをチェックする。最大許容消費率315が指定されていないか、予測消費率がそれより大きくなければ、アクセス制御リスト強制手段は消費が許される旨を表示して終了する。予測消費率が大きければ、アクセス

制御リスト強制手段は、ステップ530で、この動作を実行するのに必要な遅延を、（実際使用量300 + REQAMT）／最大許容消費率315 - （現時点 - コード開始時点395 - COMPT）に従って計算し、計算された遅延時間だけ消費が遅延しなければならない旨を表示するとともに、必要な遅延時間を戻す。

【0025】リソースが消費された後、ステップ550で、アクセス制御リスト強制手段はリソース消費量（CONSAMT）を指定するパラメータを使用して呼び出される。呼び出されたアクセス制御リスト強制手段は、実際使用率325と実際使用量330を更新する。

【0026】さらに、同じコードについて、受け取りシステムで、異なったユーザには異なったリソースと許容条件を割り振ることができる。それは、コードがインストールされる時点では、アクセス制御リスト強制手段により、異なったユーザへ与えられた特権に注目して、コードに許されたリソースと許容条件に特権を組み合わせることによって行うことができる。この場合、リソースと許容条件の集合は、各ユーザごとに別々に記憶されることが必要であろう。コードが実行されているときには、リソースと許容条件はユーザ・ベースで強制されることになろう。他方、コードの実行中にリソースと許容条件を決定する場合は、アクセス制御リスト強制手段により、異なったユーザへ与えられた特権に注目して、コードに許されたリソースと許容条件に特権を組み合わせることによって決定することができる。

【0027】図7は、クライアント・システムで行われる完全性検査、実行、および強制の初期設定動作を示す疑似コードである。今までに説明した各種のテーブル、リスト、フラグその他のデータ構造はクライアント・システムのメモリ（たとえば、揮発性ランダム・アクセス・メモリ、ディスク、またはこれらの組み合わせ）の中でインスタンス化されることに注意されたい。前述したように、アクセス制御リスト強制手段とアクセス制御リスト管理手段は、プログラム・コードとして実施するのが望ましい。これらのプログラム・コードは、クライアント・システムのオペレーティング・システムへリンクされるか組み込まれる。図8はアクセス制御リスト管理手段の疑似コードを示す。図9はアクセス制御リスト強制手段の疑似コードを示す。

【0028】実施例に基づいて本発明を説明してきたが、当業者による各種の変更と改善が可能であると思われる。したがって、本発明の実施例は単なる例であって、制限的に解釈されてはならないことを理解すべきである。本発明の範囲が請求範囲によって限定されることはいうまでもない。

【0029】まとめとして、本発明の構成に関して以下の事項を開示する。

（1）プログラム・コードの配布方法であって、信頼さ

れる第三者の証明を受け取りシステムへ提供するステップを含み、上記プログラム・コードの検査された無害の動作に必要なリソースと許容条件を、上記信頼される第三者の証明の中にコンピュータで読み取り可能に記述したことを特徴とする、プログラム・コードの配布方法。

(2) 上記(1)において、上記証明をプログラム・コードと共にカプセル化するステップを含む、プログラム・コードの配布方法。

(3) 上記(1)において、上記受け取りシステムが上記証明を読み取るステップと、上記証明中に指定された許容条件を超過しないように上記受け取りシステムのリソースと許容条件を割り振るステップとを含む、プログラム・コードの配布方法。

(4) 上記(2)において、上記証明に関連してユーザが選択したオプションに従って、上記受け取りシステムのリソースに対する上記プログラム・コードのアクセスと許容条件を否定または肯定するステップを含む、プログラム・コードの配布方法。

(5) 上記(1)において、コンピュータで読み取り可能に記述した上記リソースと許容条件が暗号形式で上記受け取りシステムへ提供されることを特徴とする、プログラム・コードの配布方法。

(6) 上記(1)において、暗号化された検査データが上記証明の中に含まれており、上記受け取りシステムが上記検査データを解読して上記リソースと許容条件の記述の完全性を検査することを特徴とする、プログラム・コードの配布方法。

(7) 上記(3)において、必要なリソースの記述の中に、上記プログラム・コードによって使用される各リソースの量と、そのリソースの最大許容消費率の両方が記述されていることを特徴とする、プログラム・コードの配布方法。

(8) 上記(3)において、必要な許容条件の記述の中に、上記プログラム・コードによってアクセスされる上記受け取りシステムの特定の機能が記述されていることを特徴とする、プログラム・コードの配布方法。

(9) 上記(1)において、上記プログラム・コードの機能性が、上記第三者による検査に従って上記第三者の証明の中に記述されていることを特徴とする、プログラム・コードの配布方法。

(10) 上記(1)において、上記プログラム・コードが、サーバからプログラム・オブジェクトとしてダウンロードされるアプレットであることを特徴とする、プログラム・コードの配布方法。

(11) 上記(1)において、上記受け取りシステムの中で上記プログラム・コードの異なるユーザに対しては異なるリソースと許容条件を割り振ることを特徴とする、プログラム・コードの配布方法。

(12) 上記(11)において、上記異なるユーザに許されるリソースと許容条件の組が、上記プログラム・

コードのインストール時に決定されることを特徴とする、プログラム・コードの配布方法。

(13) 上記(11)において、上記異なるユーザに許されるリソースと許容条件の組が、上記プログラム・コードの実行時に決定されることを特徴とする、プログラム・コードの配布方法。

(14) プログラム・コードの配布方法であって、信頼される第三者によるプログラム・コードの証明の中に、該プログラム・コードの検査された無害の動作に必要なリソースと許容条件をコンピュータで読み取り可能に記述したものを入れ、その証明を上記プログラム・コードと共にカプセル化して暗号形式で受け取りシステムへ提供するステップと、上記受け取りシステムが上記暗号形式の証明を読み取るステップと、上記読み取られた証明の完全性を上記受け取りシステムによって決定するステップと、上記完全性が検査された後に、上記証明の中で指定された許容条件を超過しないようにユーザが選択したオプションに従って上記受け取りシステムのリソースと許容条件を割り振るステップと、上記割り振りに従って上記プログラム・コードを実行するステップとを含む、上記必要なリソースの記述の中には、上記プログラム・コードによって使用される各リソースの量とその最大許容消費率との両方が記述され、上記許容条件の記述の中には、上記プログラム・コードによってアクセスされる上記受け取りシステムの特定の機能が記述されていることを特徴とする、プログラム・コードの配布方法。

(15) コンピュータ・システムへプログラムとデータを移入する移入装置と、上記コンピュータ・システムの動作を制御するオペレーティング・システムと、コードの検査された無害の動作に必要なリソースをコンピュータで読み取り可能に記述したものを、上記データから抽出してそれを所与のプログラムに関連づけるアクセス・ロジックと、該アクセス・ロジックに含まれて、上記コンピュータで読み取り可能に記述したものの完全性を示す検査データを生成する完全性検査ロジックと、上記オペレーティング・システムに接続され、上記検査データに応答して多数のリソースの各々について上記受け取りシステムの中で消費量と消費率を追跡し割り振って、上記記述の中に指定された割り振りを超過しないようにする強制ロジックと、上記割り振りに従ってプログラム・コードを実行するプロセッサと、を具備するコンピュータ・システム。

(16) 上記(15)において、ランダム・アクセス・メモリに記憶されたデータ構造が上記強制ロジックに接続されており、該データ構造の中には、消費された実際のリソースを追跡する第1のフィールドと、リソースの消費率を追跡する第2のフィールドと、上記記述から引き出されたリソース消費の限度を記憶する第3のフィールドと、上記記述から引き出されたリソース消費率の限度を記憶する第4のフィールドとが含まれていることを

特徴とする、コンピュータ・システム。

(17) 上記(15)において、上記アクセス・ロジックの中に、プログラム・コードを含むパッケージから上記記述を非カプセル化する手段と該記述を解読する手段とが含まれていることを特徴とする、コンピュータ・システム。

(18) 上記(15)において、上記強制ロジックの中に、上記記述に関連してユーザが選択したオプションに従ってコンピュータ・システムのリソースに対するコード・アクセスと許容条件を否定または肯定する手段が含まれていることを特徴とする、コンピュータ・システム。

【図面の簡単な説明】

【図1】本発明に従うコード配布・証明システムを示すブロック図である。

【図2】クライアント・システムの各部分が、実施例で説明される機能を実行するために、相互間でどのように動作するかを示す図である。

【図3】アクセス制御リスト(ACL)を示す図である。

【図4】アクセス制御リスト(ACL)強制手段によって使用されるデータ構造を示す図である。

【図5】論理リソースの許容条件がアクセス制御リスト(ACL)強制手段によってどのように強制されるかを示す図である。

【図6】物理リソースの限度がアクセス制御リスト(ACL)強制手段によってどのように強制されるかを示す図である。

【図7】クライアント・システムの完全性検査、実行、および強制的初期設定動作を疑似コードで示した図である。

【図8】アクセス制御リスト(ACL)管理手段の疑似コードを示す図である。

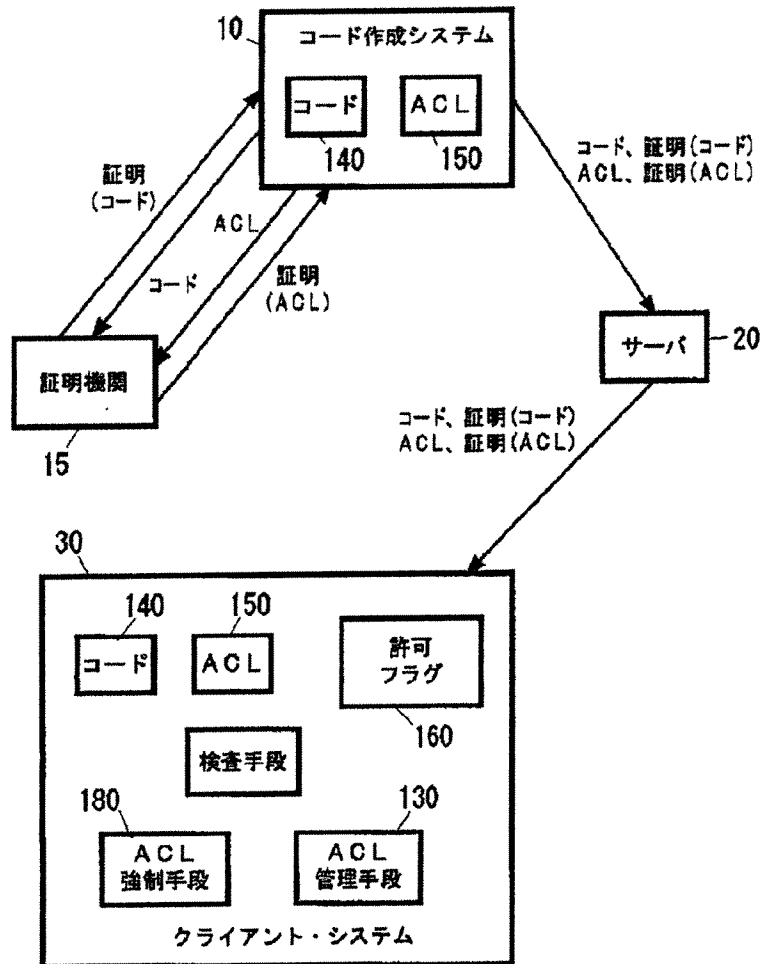
【図9】アクセス制御リスト(ACL)強制手段の疑似コードを示す図である。

【符号の説明】

10	コード作成システム
15	証明機関
20	サーバ
30	クライアント・システム
100	コード/アクセス制御リストおよび証明
110	検査手段
120	拒絶ステップ

130	アクセス制御リスト管理手段
140	コード
150	アクセス制御リスト
160	許可フラグ
162	CPUおよび関連制御ロジック
170	実行手段
180	アクセス制御リスト強制手段
190	クライアント・インタフェース
194	通信コントローラ/ネットワーク・インタフェース
196	CD-ROM/フロッピー・ディスク・サブシステム
200	物理リソース・テーブル
205	物理リソース名
210	リソース属性
215	最大許容消費率
220	最大許容量
250	論理リソース・テーブル
255	論理リソース名
260	パラメータ・リスト
265	パラメータ・エントリ
270	パラメータ・タイプ
275	パラメータ値
280	nextPE (次のパラメータ・エントリ) フィールド
300	実行時物理リソース・テーブル
305	物理リソース名
310	リソース属性
315	最大許容消費率
320	最大許容量
325	実際消費率
330	実際使用量
350	実行時論理リソース・テーブル
355	論理リソース名
360	パラメータ・リスト
365	実行時パラメータ・エントリ
370	パラメータ・タイプ
375	パラメータ値
380	nextPE (次のパラメータ・エントリ) フィールド
385	許可フラグ
395	コード開始時点

【図 1】

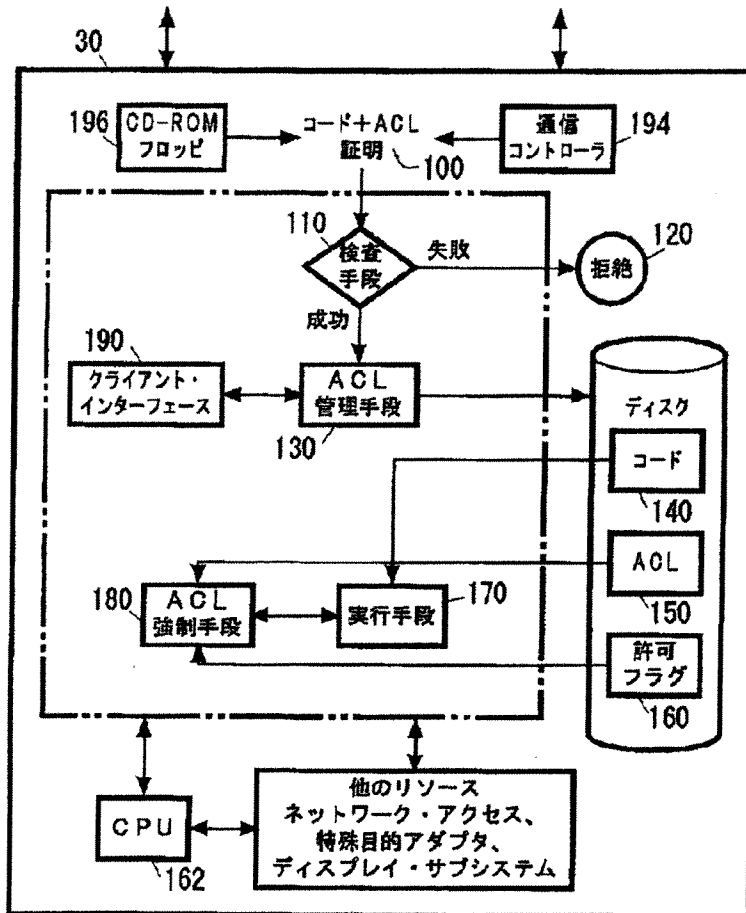


【図 8】

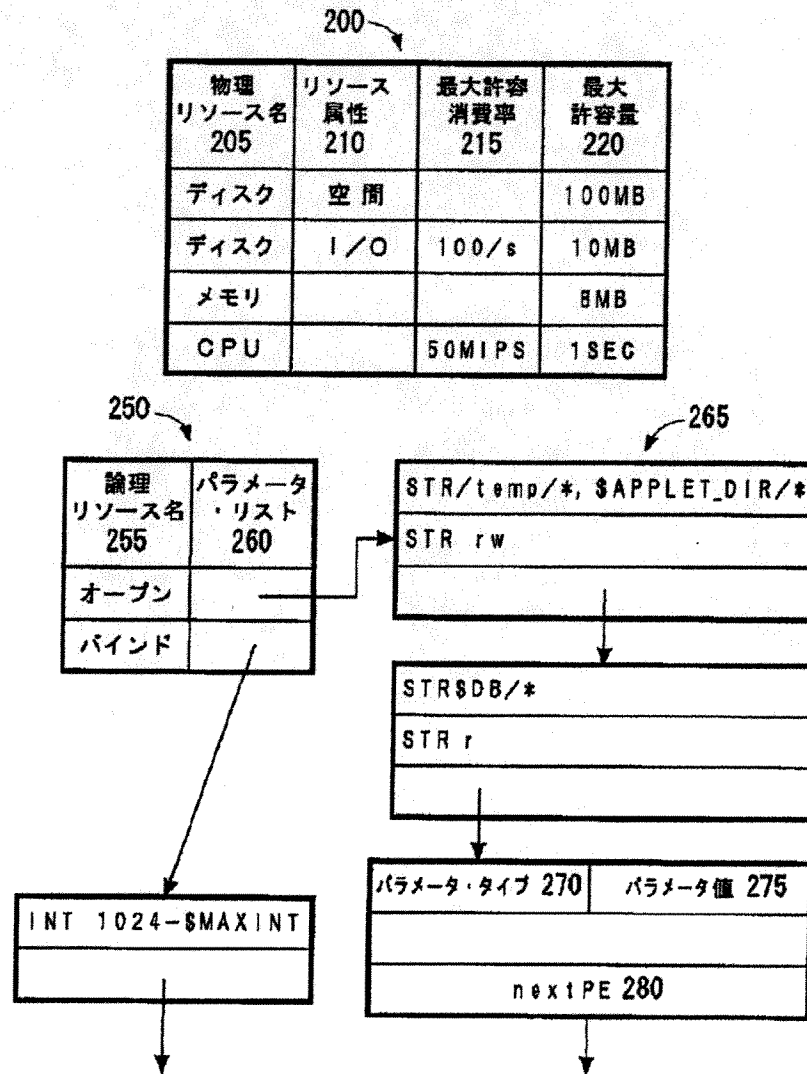
```

if 新しく受取られたコードであれば then
    コードとACLをディスク上に記憶；
    プログラムのACLを表示；
while まだクライアント入力があれば do
    クライアント入力を読取る；
    適切な許可フラグを変更する；
    変更された許可フラグをディスク上に記憶；
end
  
```

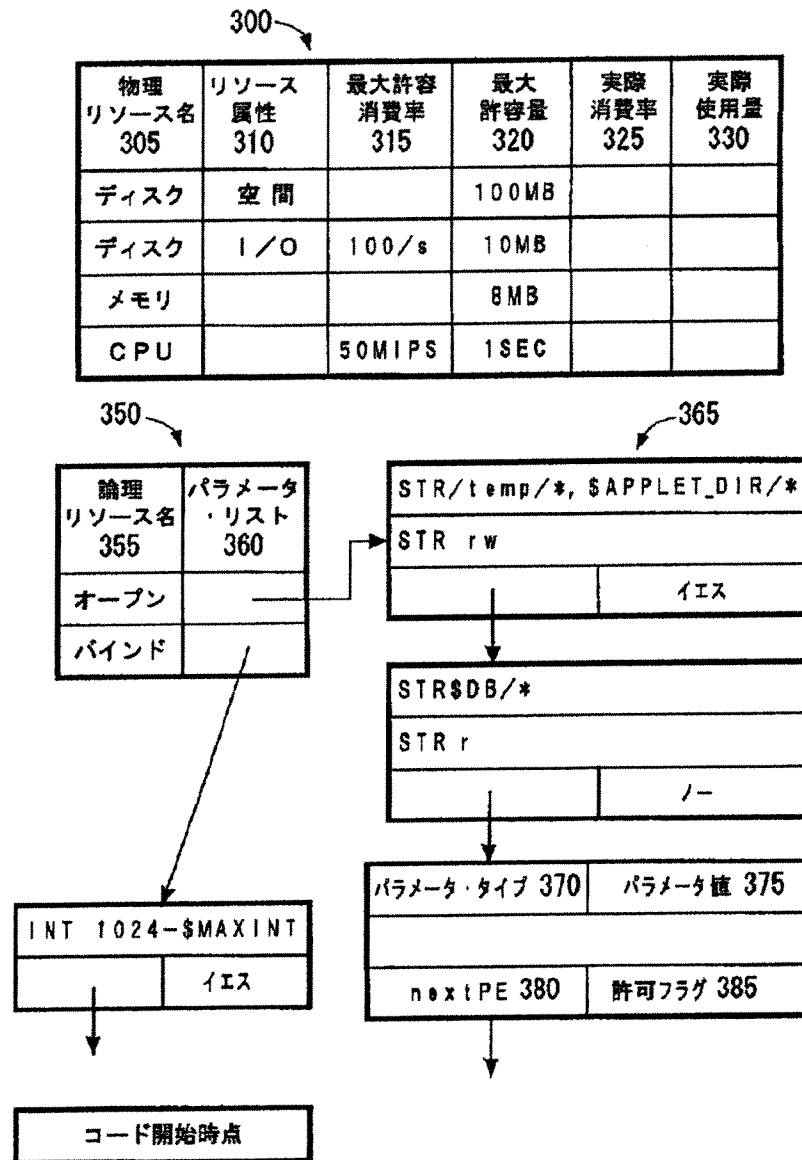
【図 2】



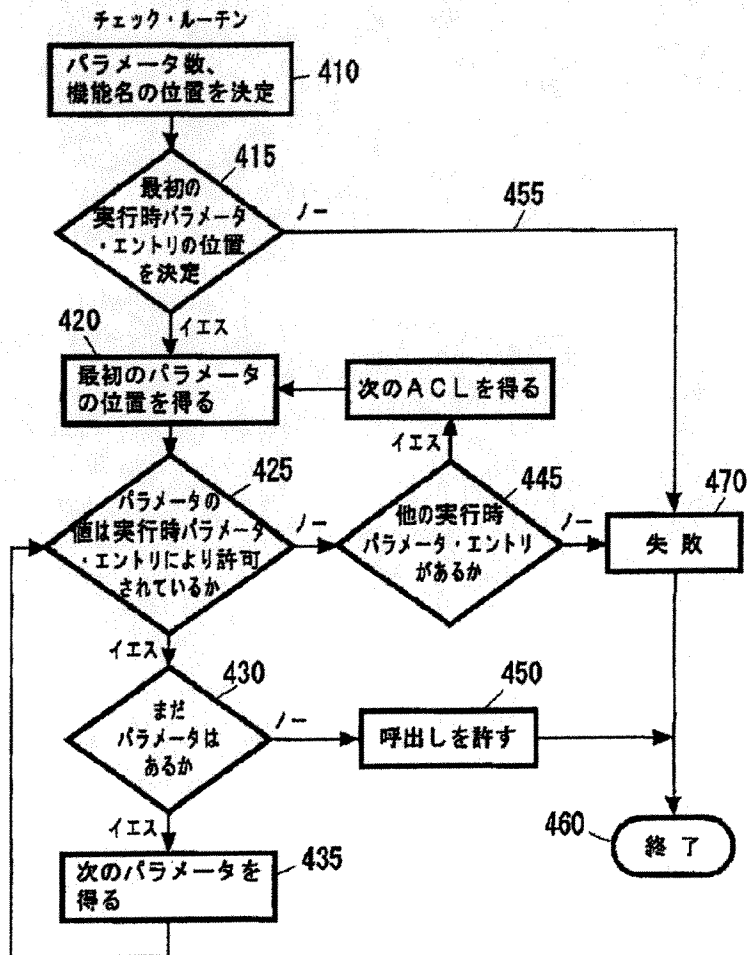
【図 3】



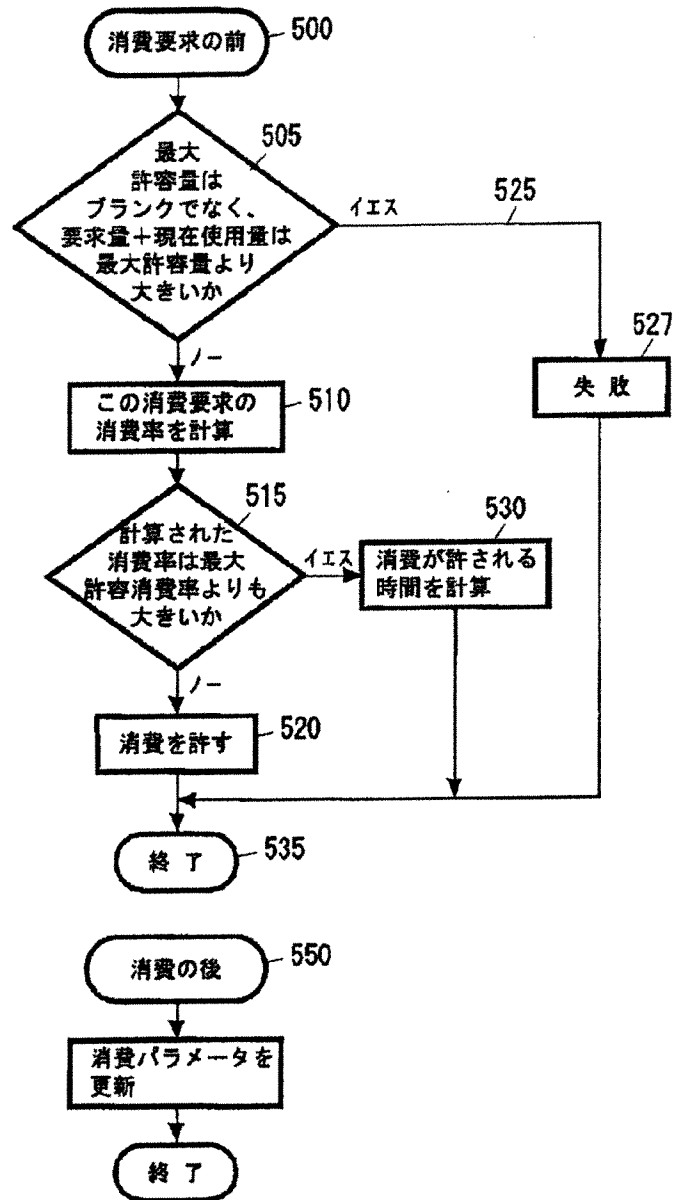
【図 4】



【図5】



【図 6】



【図7】

コードのインストール：

プログラム・コード、ACLおよび暗号ハッシュを受取る；
第三者の公用キーを使用して解読；
コードとACLからハッシュを計算；
if 計算されたハッシュが解読されたハッシュと等しくなければ
then コードを拒絶；
exit；
ACL管理手段を呼出す；

コードの実行：

リソースがアクセスされるときアクセス検査へのトラップを
コード中に挿入；
変更されたコードを実行；

アクセス検査：

ACL強制手段を呼出す；
if アクセスが許されれば then
継続；
else if アクセスが遅延されれば then
プログラムを遅らせる；
else if アクセスが拒否されれば then
プログラムを停止する；

【図9】

論理リソース・アクセスのチェック：

```

この論理リソースの実行時パラメータ・エントリの位置を決定；
while まだ実行時パラメータ・エントリがあれば do
  while まだパラメータがあれば do
    if パラメータへのアクセスが許されなければ
      goto getNextRPE；
    end
    アクセス許可を戻す；

    getNextRPE： 次の実行時パラメータ
                  ・エントリを得る；
  end
  アクセス拒否を戻す；

```

物理リソース・アクセスのチェック：

```

実行時物理リソース・テーブルの中でこの物理リソースの
エントリの位置を決定；
if 最大許容量がblankでなく、要求量+使用量が
  最大許容量より大きければ then
  アクセス拒否を戻す；
  予測消費率を計算；
  if 予測消費率が最大許容消費率よりも大きければ then
    必要な遅延を計算；
    アクセス遅延と必要な遅延を戻す；
  else
    アクセス許可を戻す；
  end
end

```

フロントページの続き

(72) 発明者 ラジェヴィ・ラマスワミ
 アメリカ合衆国10562、 ニューヨーク州
 オッシニング クロトン アヴェニュー
 52 アパートメント 6C

(72) 発明者 ディンカール・シタラム
 アメリカ合衆国10598、 ニューヨーク州
 ヨークタウン ハイツ セス レーン
 525

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-111833

(43)Date of publication of application : 28.04.1998

(51)Int.Cl. G06F 12/14
G06F 12/00

(21)Application number : 08-265746

(71)Applicant : HITACHI LTD

(22)Date of filing : 07.10.1996

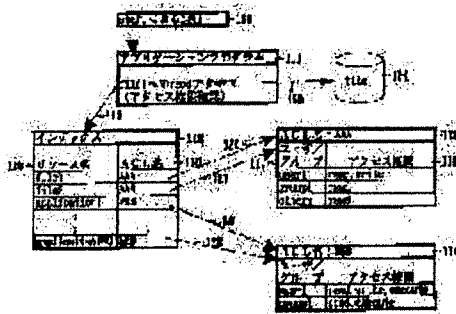
(72)Inventor : HOSAKA TAKASHI

(54) ACCESS RIGHT MANAGEMENT SYSTEM

(57)Abstract:

PROBLEM TO BE SOLVED: To enable a large-scale computer system which has many resources to flexibly and easily manage rights to access by registering the relation between an access control list(ACL) and resources in a table and then performing operations for maintenance and reference.

SOLUTION: An application program 111 is started by execution 110 by a user user1. This application program 111 gains access to a file 123 named file1, but before the access, it is confirmed whether or not there is a right to gain read access to a user1's file1. For the purpose, indexes 112 are referred to and the resource file1 is retrieved. When it is found, a corresponding ACL name AAA is obtained and the entity 113 of the ACL is obtained from it. The user user1 is retrieved from it to know that the user1 has a read right among its own access rights. Here, the read access is actually gained.



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平10-111833

(43) 公開日 平成10年(1998) 4月28日

(51) Int.Cl.*	識別記号	F I
G 0 6 F 12/14	3 1 0	G 0 6 F 12/14 3 1 0 K
12/00	5 3 7	12/00 5 3 7 A

審査請求 未請求 請求項の数 5 O L (全 5 頁)

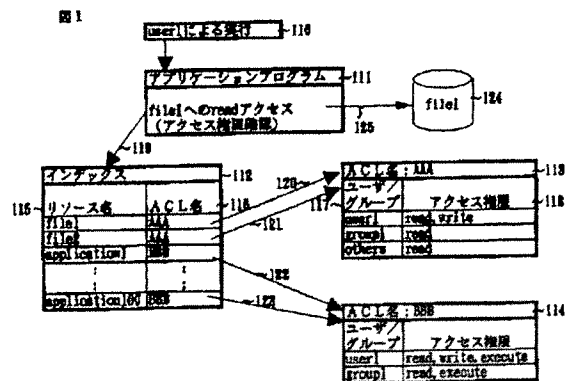
(21) 出願番号	特願平8-265746	(71) 出願人	000005108 株式会社日立製作所 東京都千代田区神田駿河台四丁目6番地
(22) 出願日	平成8年(1996)10月7日	(72) 発明者	保坂 崇 神奈川県横浜市戸塚区戸塚町5030番地 株式会社日立製作所ソフトウェア開発本部内
		(74) 代理人	弁理士 小川 勝男

(54) 【発明の名称】 アクセス権管理方式

(57) 【要約】

【課題】大規模で大量のリソースを有する計算機システムにおいてアクセス権限を柔軟な形で容易に管理することを可能にする。

【解決手段】ファイル名file1のファイル124にアクセスするアプリケーション111がアクセス権限を確認するためにインデックステーブル112を参照してfile1に対応するACLを検索しAAAというACL113を得る。ここで実際にアクセス権限の有無を確認する。インデックステーブル112上では複数のリソースを1つのACLにまとめて関連づけられている様子を示している。



【特許請求の範囲】

【請求項1】 ユーザファイル、ユーザアプリケーション、コマンド等の大量のリソースを有する計算機システムにおけるアクセス権限管理の際に任意の複数リソースをグルーピングして管理することでアクセス権限の管理の容易化を計ることができるアクセス権限管理方式。

【請求項2】 インデックスを利用することでアクセス主体のアクセス権限をリストとして管理しているアクセスコントロールリスト(ACL)とリソースとを結び付け、1つのACLで複数のリソースのアクセス権限管理を行う請求項1記載のアクセス権限管理方式。

【請求項3】 アクセス権限の検証を行うためリソースとACLの対応付けを行いアクセスの主体のアクセス権限の有無を確認する請求項1記載のアクセス権限管理方式。

【請求項4】 アクセス権限の登録・更新を行うためリソースとACLの対応付けを行い、複数のリソースのアクセス権限を一括して操作する請求項1記載のアクセス権限管理方式。

【請求項5】 新規リソース追加の際に既存リソースと同一のACLを用いて権限管理を行う場合、新たなACLを作成せずに既存リソースのACLと結びつける操作を提供する請求項1記載のアクセス権限管理方式。

【発明の詳細な説明】**【0001】**

【発明の属する技術分野】 本発明は、大規模な計算機システムにおけるアクセス権限の管理方式に関わるものである。

【0002】

【従来の技術】 従来のアクセス権限管理方式ではアクセスされる各々の対象物に対して、アクセスする主体とアクセス権限を組にしたリストを作成し、これをアクセスコントロールリスト(ACL)として実際にアクセスが行われる時にそのアクセス方法に該当するアクセス権限を有するか否かをACLを参照することにより制限できた。しかし各々の対象物毎にこのACLを作成・管理するために大量のリソースを有する大規模な計算機システムにおいては管理が難しいという従来の問題点が残っている。この問題を解決するために「セキュリティ管理の技術、日科技連、p. 23~24」に記載されている名称規則による総称指定、グループ定義といった手段が存在していたがそれぞれ名称が一部固定されてしまう、アクセスの対象物のアクセス権限管理としては変わらず管理に労力を要するといった問題が残されている。

【0003】

【発明が解決しようとする課題】 上記の従来技術では、大規模で大量のリソースを有する計算機システムにおいてアクセス権限の管理を柔軟な形で容易にすることが望まれていた。

【0004】 本発明の目的はこのアクセス権限の管理を

柔軟な形で容易にする手段を提供することにある。

【0005】

【課題を解決するための手段】 上記の目的はアクセスコントロールリスト(ACL)とリソースの関連をテーブルに登録し、この上で保守、参照といった操作を行う手段を提供し、また、必要により任意の複数リソースからなるグループを作成し、これからある1つのACLを参照する手段を提供することによって達成される。

【0006】

【発明の実施の形態】 以下、本発明の一実施例について図面で説明する。

【0007】 図1は本発明によるアクセス権限管理方式を用いた場合のアプリケーションによるアクセス権限の確認方式を示す図である。ユーザuser1による実行110によりアプリケーションプログラム111が起動される。アプリケーションプログラム111ではfile1というファイル124へのreadアクセスがある。このファイルへのreadアクセスの前にuser1のfile1に対するreadアクセス権の有無確認を行う。そのためにまず、インデックス112を参照する。ここでリソース名file1を検索する。見つかったら対応するACL名AAAを取得する。この得られてACL名AAAからACLの実体113を得る。この中でユーザuser1を検索しuser1の持つアクセス権限にread権のあることが分かる。ここでuser1によって起動されたアプリケーションプログラム111にfile1へのreadアクセスが許され、実際にfile1に対してreadアクセスが行われる。

【0008】 図2は本発明によるアクセス権限管理方式におけるACL操作コマンドの操作を示す図である。ACL操作コマンド210はアクセス権限の確認に用いるインデックス211とACLの実体212、213の操作する手段を提供している。インデックスの操作としてリソース名214の登録、リソース名214とACL名215との関連づけを行う。ACLの操作としてはACLの作成、ユーザ/グループ216の追加・削除、アクセス権限217の変更等がある。

【0009】 これを用いることでACLによる権限の管理を行う。

【0010】 図3は図1におけるリソースとACLの関連づけの概念を示す図である。リソース群に登録したfile1からfile3は同じACL: AAAと関連づけられている。これでACL: AAAを操作することでfile1からfile3のACLを一括して操作することが可能になる。同様にapplication1からapplication100はACL: BBBに関連づけられている。従ってACL: BBBを操作することでapplication1からapplication100のACLを一括して操作できることになる。file10とapplication200はAC

L:CCCと関連づけられている。このように各ACLに関連づけられるリソース名はユーザの任意の名称での登録が可能となっている。また任意の組み合わせも可能となっている。

【0011】図4は図2におけるACL操作コマンドの操作例を示す図である。411はユーザのアクセス権限登録の例を示す。ACL操作コマンドにパラメタとしてユーザ名user1、user1に与える権限としてread、そしてACL名AAAを指定する。この操作によりACL:AAAにuser1のread権限が登録される。412にACLに対するリソースの関連づけ例を示す。

【0012】ACL操作コマンドのパラメタとしてACL名、リソース名の列を指定する。

【0013】この操作によりACL:AAAとリソース1、リソース2、リソース3とが関連づけされる。

【0014】図5は図1におけるアクセス権限検証のフローチャートである。user1によりfile1に対するread操作が開始する(510)。まずインデックスよりリソース名file1を検索する(511)。対応するACL名AAAを取得する(512)。ACL AAAを開く(513)。ユーザuser1をACL上で検索する(514)。user1が見つかったらread権限の有無を確認する(515)。もし有れば実際にファイルアクセスを実行(517)してfile1へのread処理操作を終了する(518)。もしuser1にread権限がなければファイルアクセスエラー発生(516)としてfile1へのread処理操作を終了する(518)。

【0015】

【発明の効果】以上説明したように、本発明によれば、大量のリソースを有する大規模な計算機システムにおいて柔軟で且つ容易にアクセス権限の管理を行うことを可能にする。

【図面の簡単な説明】

【図1】本発明によるアクセス権限管理方式を用いた場合のアプリケーションによるアクセス権限の確認方式を示す図である。

【図2】本発明によるアクセス権限管理方式におけるACL操作コマンドの操作を示す図である。

【図3】図1におけるリソースとACLの関連づけされた様子を示す図である。

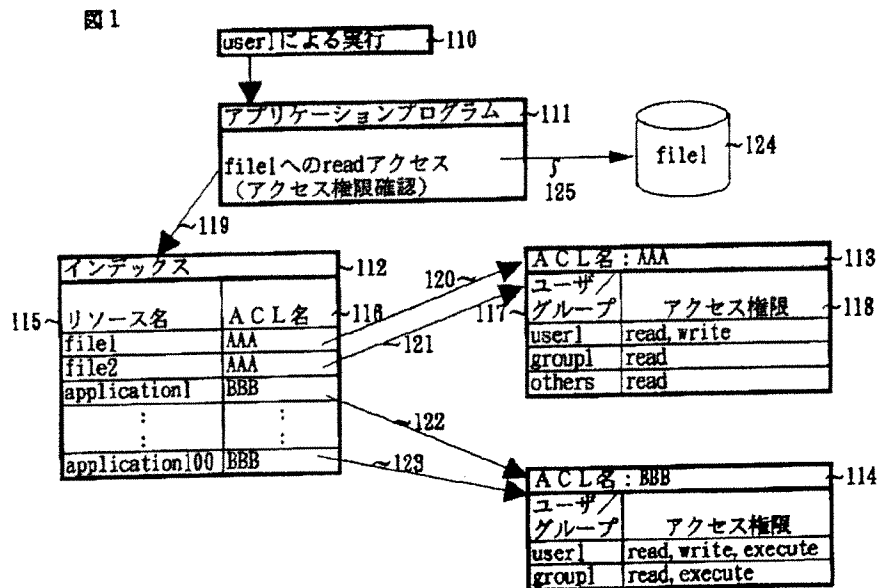
【図4】図2におけるACL操作コマンドの操作例を示す図である。

【図5】図1におけるアクセス権限検証のフローチャートである。

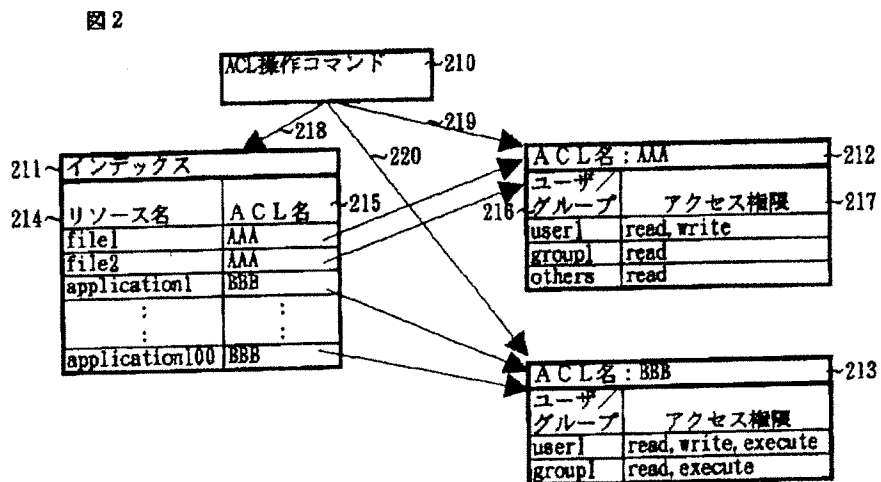
【符号の説明】

10…アプリケーションプログラム、11…リソース、12…インデックステーブル、13…ACL (ACL名: ACL1)、14…ACL (ACL名: ACL2)、15…リソース名欄、16…ACL名欄、17…ユーザ/グループ欄、18…アクセス権限欄、19…インデックス上のACL名: ACL1とACLの実体の関連づけ、20…インデックス上のACL名: ACL2とACLの実体の関連づけ、21…アプリケーションプログラムからアクセス権限確認、22…アプリケーションプログラムからリソースへのアクセス、23…ACL操作コマンドによるインデックスの操作、24…ACL操作コマンドによるACL1の操作、25…ACL操作コマンドによるACL2の操作、110…user1による実行操作、111…アプリケーションプログラ、112…インデックステーブル、113…ACL (ACL名: AAA)、114…ACL (ACL名: BBB)、115…リソース名欄、116…ACL名欄、117…ユーザ/グループ欄、118…アクセス権限欄、119…アプリケーションプログラムからアクセス権限確認、120…インデックス上のACL名: AAAとACLの実体の関連づけ、121…インデックス上のACL名: AAAとACLの実体の関連づけ、122…インデックス上のACL名: BBBとACLの実体の関連づけ、123…インデックス上のACL名: BBBとACLの実体の関連づけ、124…ファイル (ファイル名: file1)、125…アプリケーションプログラムからファイルへのアクセス、210…ACL操作コマンド、211…インデックステーブル、212…ACL (ACL名: AAA)、213…ACL (ACL名: BBB)、214…リソース名欄、215…ACL名欄、216…ユーザ/グループ欄、217…アクセス権限欄、218…ACL操作コマンドによるインデックステーブルに対する操作、219…ACL操作コマンドによるACL AAAに対する操作、220…ACL操作コマンドによるACL BBBに対する操作、311…リソース群、312…ACL群、313…リソース file1、314…リソース file2、315…リソース file3、316…ACL AAA、411…ユーザのアクセス権限登録例、412…ACLに対するリソースの関連づけ例

【図1】



【図2】



【図4】

図4

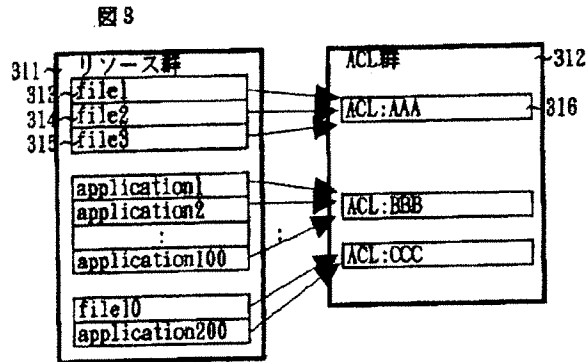
ユーザのアクセス権限登録例

411 ACL操作コマンド user1:read AAA

ACLに対するリソースの関連付け例

412 ACL操作コマンド AAA リソース1 リソース2 リソース3

【図3】



【図5】

